Design Wave 設計コンテスト 2006 関連企画

デバイスの記事

2次元積符号用繰り返し型デコーダの性能評価

林 輝彦

多くのエンジニアにとって，エラー訂正はなかなかとっつきにくい技術です．エラー訂正技術をきちんと理解するには，やはり自分で「動かしてみる」ことがいちばんでしょう．本稿では，*SN*比と*BER*（ビット・エラー率）の関係を調べた後，表計算ソフトウェアを活用して繰り返し型デコーダのアルゴリズムを評価してみます．　（筆者）

Design Wave設計コンテストも今年で6年目になります．今年の課題は**図1**に示すような，2次元積符号用繰り返し型デコーダの設計です．ベイズの定理を使いながら導き出されたデコーダ回路例も課題の出題の中で提案されていますし，課題の検証用に*SN*比の異なる信号データも用意されています．まずはここに提案されたデコーダを動作させて，エラー訂正が機能することを自分で確かめてみると，比較的速く理解が進むと思います．

ベイズの定理はかなり古くからある確率論の一つですが，近年はいろいろな分野で応用されているようです．インターネットを使ってベイズの定理を検索すると，ベイズの定理のいろいろな例題，応用例を議論しているWebサイトが多数あります．

1．*SN*比と*BER*の関係を調べてみる

今回の課題では‘0’と‘1’のデータをBPSK（binary phase shift keying）で無線信号（電波）に変調することを想定しています．まずはエラー訂正機能を使わずに，受信したBPSK信号を単に0よりも大きいか小さいかだけで判別した場合に，信号の*SN*比に従ってどの程度まちがって判別されるのかを調べてみます．信号データとしては，用意されている*SN*比の異なる5種類のパターンを用います．後でエラー訂正機能を備えたデコーダを用いて復調した結果と比べることで，エラー訂正機能の実力を実感できるでし

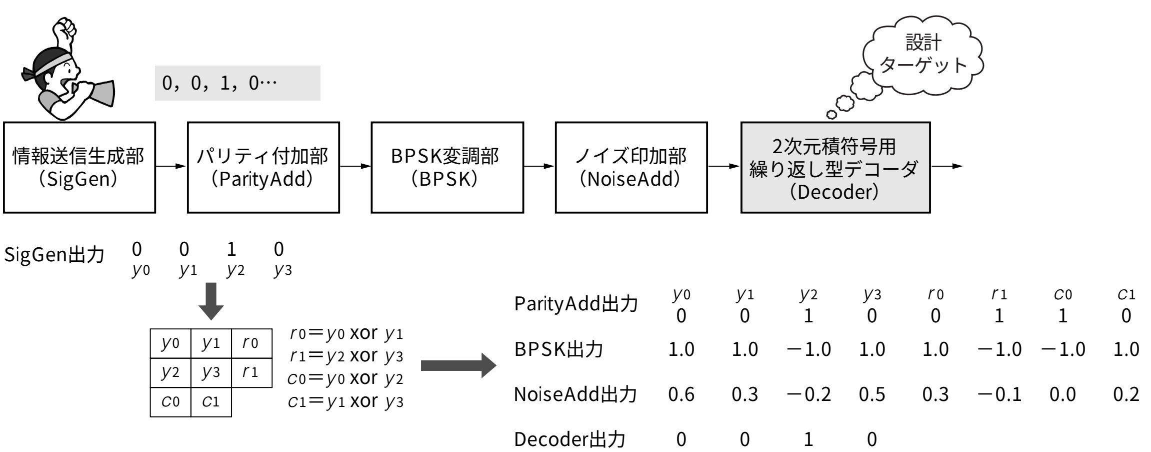


図1　システム・ブロック図[1]

設計仕様で示された図．設計仕様の詳細は，本誌2005年11月号，pp.131-140の記事で解説されている．また本誌のWebサイト（http://www.cqpub.co.jp/dwm/contest/）でも公開されている．

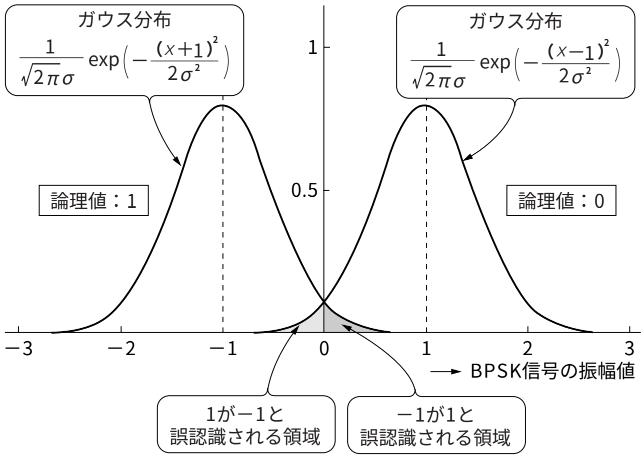


図2　ノイズの加わったBPSK信号の振幅値の分布
信号にノイズが加わることで，BPSK信号の取る値は＋1と−1を中心とする「山」の分布となる．また，加わるノイズが増加するに従ってこの山はより広がった分布になる．

ょう．

論理値の‘0’を信号としては＋1へ，論理値の‘1’を信号としては−1へ割り付けたBPSK信号に，ランダムなノイズが混入したとします．ここで想定するノイズは，ノイズの振幅値の分布がガウス分布に従う加法性白色ガウス・ノイズです．信号にノイズが混入していない状態では，BPSK信号の値は＋1か−1の二つの値しかありません．しかし，信号にノイズが加わることで，BPSK信号のとる値は＋1と−1を中心とする「山」の分布になります．また，加わるノイズが増加するに従って，この山はより広がった形になります（**図2**）．分布の裾野の部分で0と交差した部分が，本来の値とは逆の値に認識されてしまう部分となります．

BPSK信号に含まれる信号（S）とノイズ（N）のパワーを考えてみます．課題の説明の中にもあったように，信号のパワーはつねに1になっています．正規分布に従うランダムなノイズのパワーは，その分散σの2乗に等しいので，信号のSN比（SNR）は以下のようになります．

$$SNR = \frac{1}{\sigma^2}$$

信号の値が0を超え，逆の値に認識されてしまう比率は，1または−1を中心とするガウス分布の曲線を0から負の無限大まで（あるいは正の無限大から0まで）積分すればよく，以下のようになります．

$$\text{誤って認識される比率} = \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{0}\exp\left(-\frac{(x-1)^2}{2\sigma^2}\right)dx$$

この積分は一見，簡単そうですが，実際のところ初等関数では表現できません．誤差補関数$erfc$を使うことで以下のように表すことができます．

$$\text{誤って認識される比率} = \frac{1}{2}\,erfc\sqrt{\frac{SNR}{2}}$$

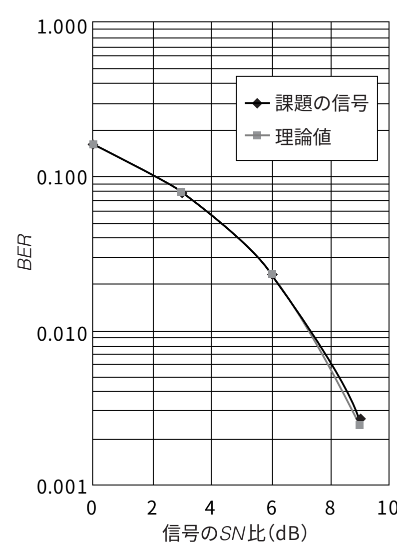
$erfc$は，統計関連の解析ではポピュラな関数です．一般的な表計算ソフトウェア（例えば米国Microsoft社のExcel）でも組み込み関数の一つとして利用できます．

$$erfc(x) = \frac{2}{\sqrt{\pi}}\int_{x}^{\infty}\exp\left(-u^2\right)du \qquad \text{（誤差補関数）}$$

伝送された全ビット数に対してまちがって伝送されたビット数の比率をBER（bit error rate）とすれば，今回のBPSK信号において，エラー訂正を行わない場合のBERとSN比の関係は以下のようになります．

図3
課題で用意された信号の*BER*特性と*erfc*関数による理論値
課題で用意された5種類のSN比の異なる信号について，y_0，y_1，y_2，y_3の値で0を境界として識別するだけではまちがって受信されてしまう個数を数えてBERを算出したもの（10000ビットのデータに対する実測値）と，ガウス分布に従った理論値の比較．

信号のSN比(dB)	課題の信号（総ビット数：10,000）誤って認識されたビット数	課題の信号（総ビット数：10,000）BER	理論値 $BER = \frac{1}{2}erfc\sqrt{\frac{SNR}{2}}$
0	1626	0.1626	0.1587
3	777	0.0777	0.0789
6	232	0.0232	0.0230
9	27	0.0027	0.0024
100	0	0	（$erfc$計算不能）



$$BER = \frac{1}{2} erfc \sqrt{\frac{SNR}{2}}$$

課題で用意された5種類の*SN*比の異なる信号について，y_0，y_1，y_2，y_3の値で0を境界として識別するだけでは，まちがって受信されてしまう個数を数えて*BER*を算出したもの(10000ビットのデータに対する実測値)と，この式のガウス分布に従った理論値を比較した結果が**図3**です．

2．表計算ソフトでアルゴリズムを理解する

課題の中で提案されている繰り返し型デコーダは，たいへんシンプルなデータパスの形をとっています．これだけのデータパスにシリアル-パラレル変換した信号を流すだけでエラー訂正ができてしまうことはちょっと意外なくらいです．データパスの中で行っている演算は加算，大小比較，正，負の符号の入れ替えといった比較的単純な演算ですから，表計算ソフトウェアの中でセル中に式を記述して展開するだけで簡単にデコーダのアルゴリズムを動かしてみることができます．

図4は，Excelの表の中で繰り返し型デコーダを実現したようすです．

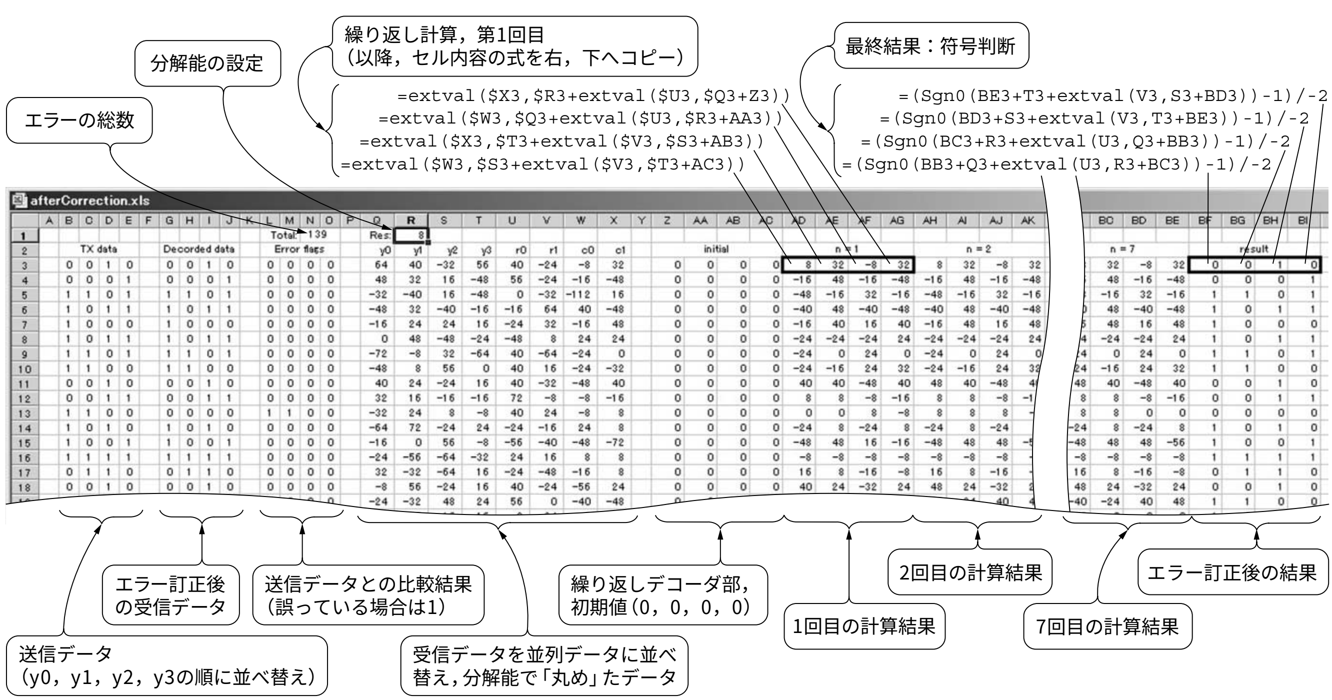
用意されている5種類の信号をこの表計算ソフトウェアの中でデコードした場合，*SN*比と*BER*の関係がどのようになるかを示したものが**図5**です．エラー訂正の機能がうまく働いて，先のエラー訂正を使用しない場合と比べると格段に*BER*が低くなっていることがわかります．

3．アルゴリズムの試行

より高速で，かつ規模の小さい回路を作るには，

(1) 目的とする処理の実行に適した，より優れたアルゴリズムを考案する

(2) アルゴリズムをハードウェアの実現方法に合わせて最



外部値計算：extval(A，B)についてはVisual Basicのマクロとして定義

```
Function extval(ar0, ay1)
    extval = Sgn0(ar0 * ay1) * Min0(Abs(ar0), Abs(ay1))
End Function
```

```
Function Min0(in1, in2)
    If in1 < in2 Then
        Min0 = in1
    Else
        Min0 = in2
    End If
End Function
```

```
Function Sgn0(in1)
    If (in1 >= 0) Then
        Sgn0 = 1
    Else
        Sgn0 = -1
    End If
End Function
```

図4　Excel上に実現した2次元積符号用繰り返し型デコーダ

適化する

の二つのアプローチが基本です．今回の2次元積符号用デコーダについて，出題時に提案された繰り返し型デコーダとは異なるアルゴリズムによる設計も，多数応募されることを期待しています．

● データ幅の検討

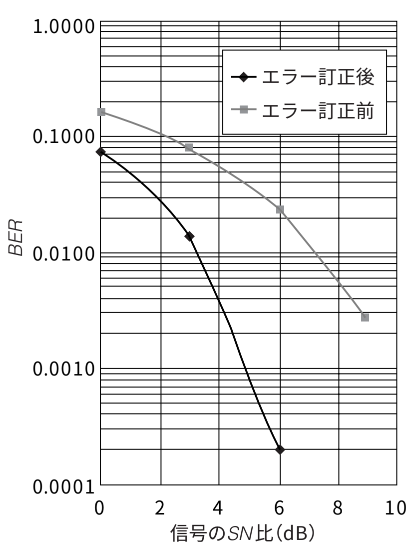
アルゴリズムを決定した後，ハードウェア化する過程で取りうる選択肢も多数あります．

処理系の中で使用するデータのビット幅の選択は，そのもっとも基本的なものでしょう．例えば，繰り返し型デコーダの内部の処理を，受信信号のもともとのビット幅である8ビットよりも狭い幅で処理した場合，*SN*比と*BER*の関係がどのように変わるかを調べてみましょう．

受信データの*SN*比が変化したときの*BER*の特性に有意な差がないのなら，デコーダを構成する処理をなるべく少ないビット数で行ったほうが回路の規模（ゲート数）が小さくなることは明らかです．今回のように，ハードウェアの多くは繰り返しデータを流すためのデータパスに費やされ，データの流れや制御を司る部分のハードウェアがデータパスの部分に比べて小さいと考えられる場合，処理系のビット数を減らすことは確実にハードウェアの削減につながります．一方で，処理系のビット幅を少なくすることは，それだけ信号に含まれる量子化ノイズの量を増やすことになるので，信号の実質的な*SN*比が低下して*BER*が増加することも予想されます．

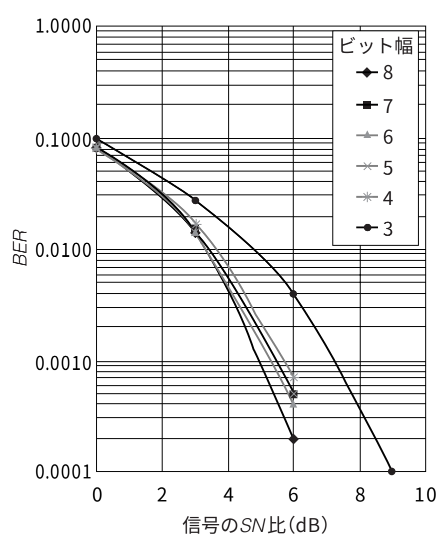
今回の課題では，検証のために20000サイクル分（10000ビットの送信データ）のデータが2の補数表現の8ビット・データとして用意されています．この8ビットのデータの範囲，－128～＋127の中で，ノイズのない元のBPSK送信データは＋32と－32に設定してあります．

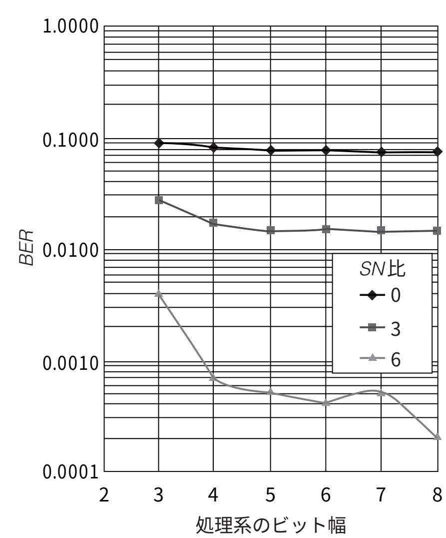
図6は，先にExcelの表で用意したデコーダのモデルに対して，2のべき乗の数（2，4，8，…）で信号の値を丸めた受信データを与え，そのときの*BER*を測定したものです．



		エラー訂正後（総ビット数：10,000）		エラー訂正前（総ビット数：10,000）
		誤り復調ビット数	*BER*	*BER*
信号のSN比（dB）	0	735	0.0735	0.1626
	3	141	0.0141	0.0777
	6	2	0.0002	0.0232
	9	0	0	0.0027
	100	0	0	0

図5
2次元積符号用繰り返し型デコーダの*BER*特性
表計算ソフトウェアExcelで求めたもの．





		処理系のビット幅/分解能					
		8	7	6	5	4	3
信号のSN比（dB）	0	0.0735	0.0741	0.0749	0.0763	0.0804	0.0919
	3	0.0141	0.0143	0.0144	0.0139	0.0169	0.0269
	6	0.0002	0.0005	0.0004	0.0005	0.0007	0.0040
	9	0	0	0	0	0	0.0001
	100	0	0	0	0	0	0

図6
ビット幅と*BER*特性
Excelの表で用意したデコーダのモデルに対して，2のべき乗の数（2，4，8，…）で信号の値を丸めた受信データを与え，そのときの*BER*を測定したもの．

例えば，受信データに対して2で丸め処理を行えば，処理系の実質的なビット幅は7ビットとして処理したことと同じになります．**図6**の結果を見ると，5ビットないし4ビットのデータ幅でも，8ビットの場合とほぼ同じ*BER*特性が得られる可能性のあることがわかります．

この例では，LSB側のデータを減らしていく方法をとっています．ただし，単純に不要となったビットを無視する方法ではありません．ビット数を減らした後の系においても，＋1，－1という中心的状態がコードの境界に来ないように「丸め」処理を行わないと，*BER*特性が悪化してしまいます．また，ノイズのピークで信号を飽和させる危険があるので今回は試してはいませんが，MSB側のデータを削減する方法もおもしろいと思います．

● EXTVALブロックの計算方法の検討

ビット幅の選択以外にも，今回の課題にはいくつか重要な検討事項があります．例えば，外部値の計算を行うEXTVALブロックの計算方法については本来，

$$Le = \ln\left(\frac{1+\exp(A+B)}{\exp(A)+\exp(B)}\right)$$

の計算を行うところを，

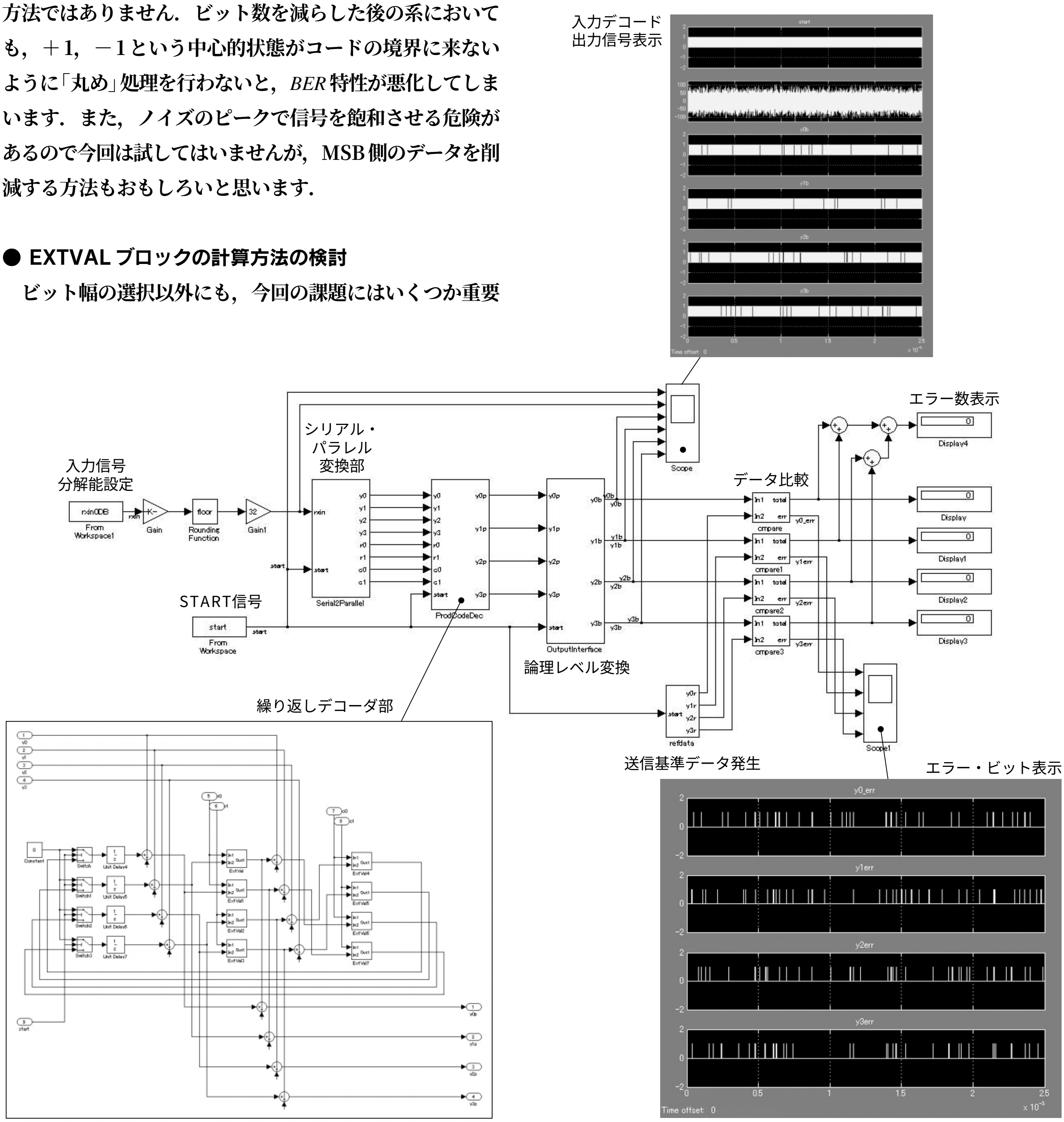


図7　MATLAB/ Simulink上に構築した2次元積符号用繰り返し型デコーダ
Simulinkによるシミュレーション結果．あたりまえだが，図5のExcelで求めた結果と完全に一致する．

$$Le \quad \mathrm{sgn}(A \cdot B)\min(|A| \cdot |B|)$$

と簡素化した計算を行っています．この簡素化によって，デコーダのエラー訂正の性能がどの程度犠牲になっているのか興味のあるところです．

また，課題で提案されている繰り返し型デコーダでは，複数の同じ回路ブロックがいくつも並列に動作しています．回路のタイミングをくふうして回路中のリソースを共用することで，大幅に回路規模を小さくすることに挑戦するのもおもしろいのではないでしょうか？

4. MATLAB/Simulinkでシミュレーション

米国MathWorks社の信号処理シュミレータ「MATLAB」は，いろいろな信号の処理に関する解析を行う環境として，現在広く使われています．とくにオプションのSimulinkを加えると，グラフィカルに各種のモデルを構築してシステム全体の動作を効率良く検証できます．

図7は，課題で示されていたデコーダをSimulink上で記述してみたものです．課題の信号をワークスペースに読み込んで復調アルゴリズムを実行し，復調後のデータと送信データを比較して誤ってデコードされたビットの数をy_0，y_1，y_2，y_3ごとにカウントします．もちろん，課題の信号を用いたSimulinkによるシミュレーション結果は，先のExcelで求めた結果と完全に一致します．

最近では，MATLAB/SimulinkとHDLシミュレータを接続して協調検証を行う環境が各社から提供されています（例えば米国Aldec社の「Active-HDL」とのインターフェースなど）．こういった環境が利用できるのであれば，HDLのコーディングの終わったブロックを順次Simulink上のモデルと入れ替えながら，Simulink上に構築した検証環境を使ってHDL部分の検証を進めることができます．

5. 設計の性能評価に関して

昨年のDesign Wave設計コンテストの課題は，FMディジタル・レシーバでした．その審査に参加させていただいた際，でき上がった設計の性能の評価をもう少し客観的に，できればなんらかの数字をもってアピールしていただけたならと感じました．FM信号を実際にFPGAで実現した回路に通して復調し，音声信号を出力させたチームもいくつかありました．しかし，FM受信機としての性能，例えば感度やひずみ率といった性能面での評価を定量的に行ったチームはほとんどありませんでした．課題用に用意された信号を復調できることが最初の目標ですが，優れた設計であることを示すためには，もう1歩踏み込んで性能を評価してほしいと思いました．

今回の設計コンテストにおいても，設計の動作速度と回路規模の計測の方法については，出題時に明確な指示が出ているので，それに従っていただければよいでしょう．そのうえで，でき上がった2次元積符号用繰り返し型デコーダの性能をなるべく客観的な指標で評価することをお勧めします．

なお，今回の記事では，*BER*評価のごく基本的なことにしか触れていません．例えば，*BER*評価の妥当性を確認するためのサンプル数やビット数といった点を議論していないので，まだ不十分であることをお断りしておきます．

それではシャノンの理論限界に挑む，最強の設計にとりかかりましょう．

参考・引用*文献

(1) * 和田知久；2次元積符号用繰り返し型デコーダ設計仕様書，pp.131-140，Design Wave Magazine，2005年11月号．
(2) 斉藤洋一；ディジタル無線通信の変復調，電子情報通信学会，1996年．
(3) 西村芳一；ディジタル・エラー訂正技術入門，CQ出版社，2004年6月．

はやし・てるひこ
(株)ソリトンシステムズ

<筆者プロフィール>
林 輝彦．根っからの「ラジオ少年」は学校卒業後，何を血迷ったか，米国系のマイクロプロセッサ会社に就職．16ビット系のコントローラ，プロセッサの開発に従事．以降，今の会社に移り，シリコン・コンパイラ，HDLシミュレータ，アナログ合成など，新しいEDAツールの普及にかかわってきたが，管理職になりつつある現在になっても設計実務から足を洗えずにいる．